Automatic parameterization of neural networks using evolutionary algorithms

https://doi.org/10.56238/sevened2024.007-089

Ádamo Henrique Rocha de Oliveira[1], Yrlles Araujo Moraes[2], Rayssa Pereira Moraes Rego Sobrinho[3], Jorleanderson Moraes Maia[4] and Marcos Silva Chaves[5]

ABSTRACT

Artificial Intelligence consists of an area where methods or systems are developed that act intelligently, approaching human behavior, in situations involving problem solving, acquisition and representation of knowledge, pattern recognition, etc. Within this context, a type of computational model that has gained prominence are the Artificial Neural Networks (ANNs), which are formed by basic blocks inspired by the biological neuron. An ANN has the ability to act in various applications, such as universal approximation of functions, process control, pattern recognition and classification, and data grouping. To meet a wide range of applications, an ANN requires the determination of a series of parameters, among them: topology, number of layers, number of neurons, activation function, training method, etc. In other words, the design of an ANN with the most appropriate configuration for each type of problem requires a series of choices, preliminary tests and experience from the designer. However, in order to avoid such choices being made empirically, it is possible to treat this parameterization as an optimization problem, allowing its resolution through the use of evolutionary algorithms, which are optimization tools developed to simulate several natural evolutionary processes. In this work, the Genetic Algorithm and Differential Evolution with binary coding were applied to automatically parameterize single-layer hidden neural networks applied in the modeling of a buck converter and in the prediction of compressive strength of self-compacting concrete (SCC) with the addition of fibers. The neural networks used were trained with the Extreme Learning Machine algorithm and the results of the simulations show that the Genetic Algorithm was the technique that presented the best performance when parameterizing the network in the modeling process of the buck converter, while the Differential Evolution combined with the binary coding GVP was the best strategy to parameterize the neural network in the process of predicting compressive strength of SCC.

Keywords: Artificial Intelligence, Artificial Neural Networks, Parameterization, Evolutionary Algorithms, Binary Coding.

[1] Highest degree of education: Master's degree
Academic institution: Federal Institute of Maranhão
[2] Highest degree of education: Master's degree
Academic institution: Federal Institute of Maranhão
[3] Highest degree of education: undergraduate
Academic institution: Federal Institute of Maranhão
[4] Highest degree of education: undergraduate
Academic institution: Federal Institute of Maranhão
[5] Highest degree of education: undergraduate
Academic institution: Federal Institute of Maranhão

INTRODUCTION

Artificial Intelligence (AI) is an area that constitutes several computational procedures whose functions performed, if a human being performed them, would be considered intelligent. The main purpose of this area is to search for methods or computational systems that possess or reinforce the capacity of intelligent behaviors of human beings, such as solving problems, acquiring and representing knowledge, recognizing patterns, among others (Lima; Pinheiro; Santos, 2016).

Intelligent systems are those that have certain capacities such as: knowledge acquisition, event planning, problem solving, information representations, knowledge storage, communication through colloquial languages and learning.

According to Lima, Pinheiro and Santos (Lima; Pinheiro; Santos, 2016), the increase in computational power that has occurred in recent decades has allowed the prominence of a line of AI research known as Machine Learning. This line of research aims to study and develop computational methods to obtain systems capable of acquiring knowledge automatically.

The main challenge of learning algorithms is to maximize the generalization capacity of their learning. Among the many promising alternatives to solve this challenge are Artificial Neural Networks.

Artificial Neural Networks (ANN) have been, over the last few years, an area of AI of great development. ANNs can be characterized as computational models capable of adapting, learning, generalizing, grouping or organizing data. This ability is acquired through models inspired by the nervous system of living beings (Lima; Pinheiro; Santos, 2016; Silva; Spatti; Flauzino, 2010).

Neural networks have some characteristics, such as non-linearity and adaptability, which allow their application in several areas. The main applications of neural networks are: universal approximation of functions, process control, pattern recognition and classification, data clustering, prediction systems, system optimization, and associative memories.

An ANN is made up of a set of basic processing units that communicate by sending information to each other through certain connections. These processing units are called neurons and are mathematical models inspired by the biological neuron (Lima; Pinheiro; Santos, 2016).

The form of interconnection of neurons in an ANN defines its architecture, the main ones being: single-layer feedforward, multi-layer feedforward, recurrent and lattice structure. Once the architecture is defined, there will also be the segmentation of the ANN into layers: the first layer receives the data to be processed; the intermediate layers treat them by extracting pertinent characteristics; and the last layer reproduces the generated results (Silva; Spatti; Flauzino, 2010).

The way an ANN acquires knowledge about a given process is known as training and can be done through several algorithms, among which the following can be mentioned: backpropagation, Newton's method, Levenberg–Marquardt and Extreme Learning Machine (Huang; Zhu; Siew, 2004; Silva; Spatti; Flauzino, 2010; Wilamowski; Irwin, 2018).

The application of an ANN to solve a given problem, in addition to the choice of topology and training method, also involves a series of determinations related to the number of neurons, activation functions, initial weights, etc. For applications in the areas of system identification and time series forecasting, additional parameters may arise, such as: number of terms applied in the input layer, types of regressors, delays of input and/or output signals, etc. Each set of parameters chosen can lead to ANN performing better or worse, depending on the type of problem being treated.

In other words, designing an ANN with the most appropriate configuration for each type of problem requires a series of choices, preliminary tests, and experience from the designer. However, in order for certain parameters not to be determined empirically, they can be treated as variables of an optimization problem, which in turn can be solved through various techniques, including evolutionary algorithms (Gaspar-Cunha; Takahashi; Antunes, 2013).

Thus, this work aims to develop an ANN design strategy, using evolutionary algorithms in some stages so that the parameterization is performed automatically, aiming at practicality in the project and obtaining the best possible performance by ANN, meeting performance criteria to be selected.

This work is organized according to the following items. Section 2 presents the theoretical basis of the topic studied. Section 3 will discuss the proposed strategy. Section 4 presents the case studies used in the work. Section 5 presents and discusses the results obtained. And finally, in section 6 are the conclusions.

THEORETICAL BACKGROUND

This section presents the main concepts related to the subjects necessary for the elaboration of the proposed strategy for parameterization of neural networks using evolutionary algorithms. Concepts about artificial neural networks, evolutionary algorithms, and binarization are discussed.

ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) are computational models inspired by the nervous system of living beings. The elemental cell of the cerebral nervous system is the neuron and its role is limited to conducting impulses under certain operating conditions (Silva; Spatti; Flauzino, 2010). Like biological neural networks, ANNs are also composed of basic units: artificial neurons (referred to as "neurons").

A neuron is an information-processing unit that is critical to the operation of a neural network (Haykin, 2001). The block diagram shown in Figure 1 shows the model of a neuron, which forms the basis for the RNA design.

Three basic elements should be highlighted in this template:

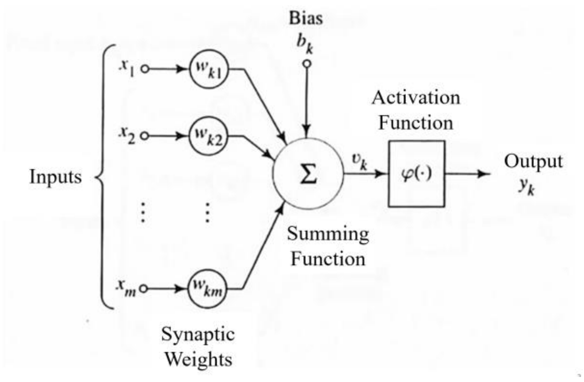
1 – A set of synapses, each characterized by a weight. Specifically, a signal x_j at the entrance of the synapse j connected to the neuron k is multiplied by the synaptic weight w_{kj}.

2 – An adder to sum the input signals, weighted by the respective synapses of the neuron; The operations described here constitute a linear combinator.

3 – An activation function to restrict the amplitude of a neuron's output.

The neuronal model in Figure 1 also includes an externally applied bias, represented by b_k. This bias has the effect of increasing or decreasing the net input of the activation function, depending on whether it is positive or negative, respectively.

Figure 1 – Nonlinear model of a neuron.



Source: Haykin (2001).

In mathematical terms, we can describe a neuron k by writing the following pair of equations:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{1}$$

and

$$y_k = \varphi(u_k + b_k) \tag{2}$$

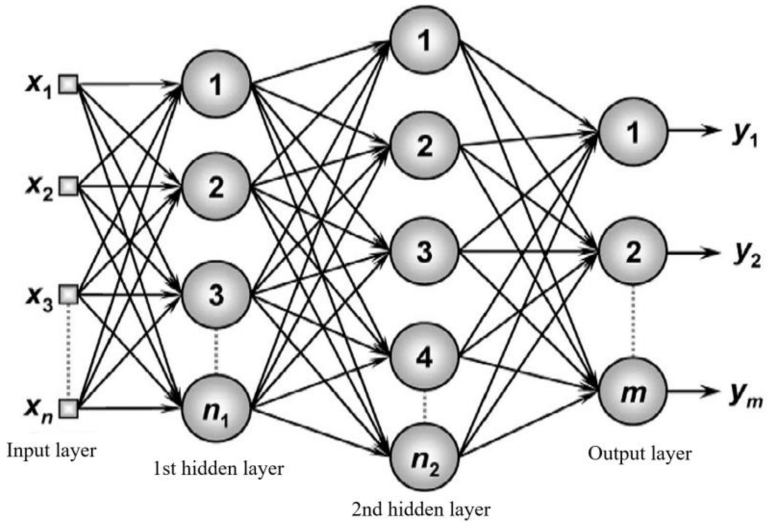
Where, x_1, x_2, ..., x_m are the input signals; w_{k1}, w_{k2}, ..., w_{km} are the synaptic weights of the neuron k; u_k is the output of the linear combiner due to the input signals; b_k it's the bias; $\varphi(\cdot)$ is the activation function; and y_k is the output signal of the neuron (Haykin, 2001).

The ways in which neurons are interconnected define the topologies or architectures of neural networks. The main neural network architectures are presented by Silva, Spatti and Flauzino (2010), such as architecture *feedforward* single-layer, architecture *feedforward* multi-layered, recurrent or feedback architecture, and lattice architecture.

Basically, an artificial neural network can be divided into three parts, called layers, which can be made up of a variable number of neurons and are named as follows: input layer, responsible for receiving information (data), signals, characteristics or measurements from the external environment; hidden layers, composed of neurons responsible for extracting the characteristics associated with the process or system to be inferred; and output layer, responsible for the production and presentation of the network's final results (Silva; Spatti; Flauzino, 2010).

Figure 2 shows an example of a neural network with the input layer, two hidden layers, and the output layer, consisting of n, n_1, n_2 and m neurons, respectively. The input layer receives the signal represented by the dataset $\boldsymbol{x} = [x_1, x_2, x_3, \dots, x_n]$, the intermediate layers process this data, and the output layer presents the final result of the network, i.e. the dataset $\boldsymbol{y} = [y_1, y_2, \dots, y_m]$. Specifically, this example network has *multi-layer* feedforward architecture.

Figure 2 – Example of a *multi-layer* feedforward network.



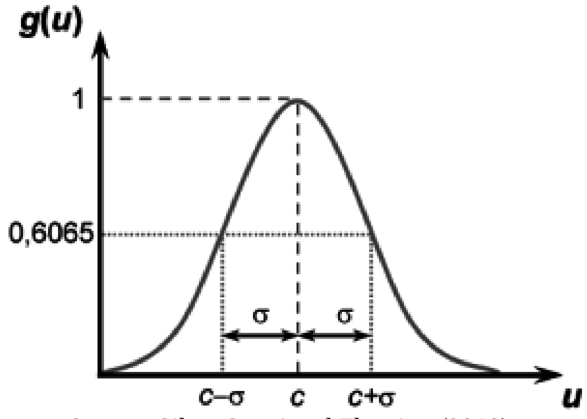
Source: Silva, Spatti and Flauzino (2010).

A given neural network architecture can have different types of topologies, which can be defined by the number of neurons used or by the different types of activation functions. Some examples of activation functions presented by Silva, Spatti and Flauzino (2010) They are: step function, sign function, symmetric ramp function, logistic function, hyperbolic tangent function, and Gaussian function. Figure 3 shows the Gaussian activation function, given by:

$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}} \qquad (3)$$

where, c is a parameter that defines the center of the Gaussian function and σ denotes the standard deviation associated with it, that is, how spread the curve is with respect to its center. Still observing the graph in Figure 3, it is possible to see that the value of the standard deviation is directly associated with the inflection point of the Gaussian function, σ^2 indicating its respective variance.

Figure 3 – Gaussian activation function.



Source: Silva, Spatti and Flauzino (2010).

Another relevant aspect of neural networks is the way knowledge is acquired and stored, that is, its training process, which consists of the application of a set of ordered steps in order to adjust the weights of the neurons (Silva; Spatti; Flauzino, 2010). Such an adjustment process, also known as a learning algorithm, then aims to tune the network so that its responses are close to the desired values. The main learning methods are: supervised, unsupervised, and reinforced.

There are several types of neural network topologies and some that deserve to be highlighted are: *Perceptrons* Single Litter, *Adaline*, *Perceptrons* Multilayer, Radial Base Function Networks, Support Vector Machines, Committee Machines, Hopfield Recurrent Networks, Self-Organizing Kohonen Networks, LVQ Networks, and ART Networks (Haykin, 2001; Silva; Spatti; Flauzino, 2010).

In the case of neural networks of the type *feedforward* Single Layer (SLFN) *Single Hidden Layer Feedforward Neural Network*), it is possible to list some training algorithms, such as *backpropagation*, Newton and Levenberg–Marquardt method (Silva; Spatti; Flauzino, 2010; Wilamowski; Irwin, 2018), but an algorithm that has been gaining prominence in the literature is the so-called Extreme Learning Machine (ELM) (Huang; Zhu; Siew, 2004). This algorithm performs the training of SLFN networks in a fast and simplified way, making use of the generalized Moore-Penrose inversion (Serre, 2002) to analytically calculate network egress weights. As presented by Huang, Zhu and Siew (2004), this technique allows to obtain the lowest norm of the weights, avoids convergence to local minima and does not require many iterative steps to obtain the best learning performance, unlike what occurs in gradient-based descending methods.

EVOLUTIONARY ALGORITHMS

Evolutionary algorithms are heuristic techniques inspired by adaptation mechanisms of living beings, as observed in nature, where certain mechanisms produce adequate responses to problems of great complexity. This effect, on which the development of evolutionary algorithms is based, occurs as a consequence of the execution of simple actions by multiple individual agents that interact with each other and with the environment, collectively producing the solution of the adaptation problem (Gaspar-Cunha; Takahashi; Antunes, 2013).

As stated by Gaspar-Cunha, Takahashi and Antunes (Gaspar-Cunha; Takahashi; Antunes, 2013), it is also worth mentioning that the class of evolutionary algorithms, in recent years, has grown in convergence with lines of research coming from the field of Operations Research, which has long been dedicated to the study of general-purpose stochastic heuristic techniques, known as metaheuristics. This allows the range of algorithms that can be applied to be even greater, increasingly enriching the development of techniques for solving complex problems.

Some examples of evolutionary algorithms are: Genetic Algorithm (Holland, 1975), Differential Evolution (Storn; Price, 1997), Evolutionary Strategies (Rechenberg, 1965; Schwefel, 1965), Genetic Programming (Koza, 1992), Ant Colony (Dorigo, 1992) and Immunoinspired Algorithms (De Castro; Timmis, 2002). In this work, the Genetic Algorithm (GA) and the Differential Evolution (DE) will be used as the algorithms applied in the automatic parameterization of neural networks.

BINARIZATION

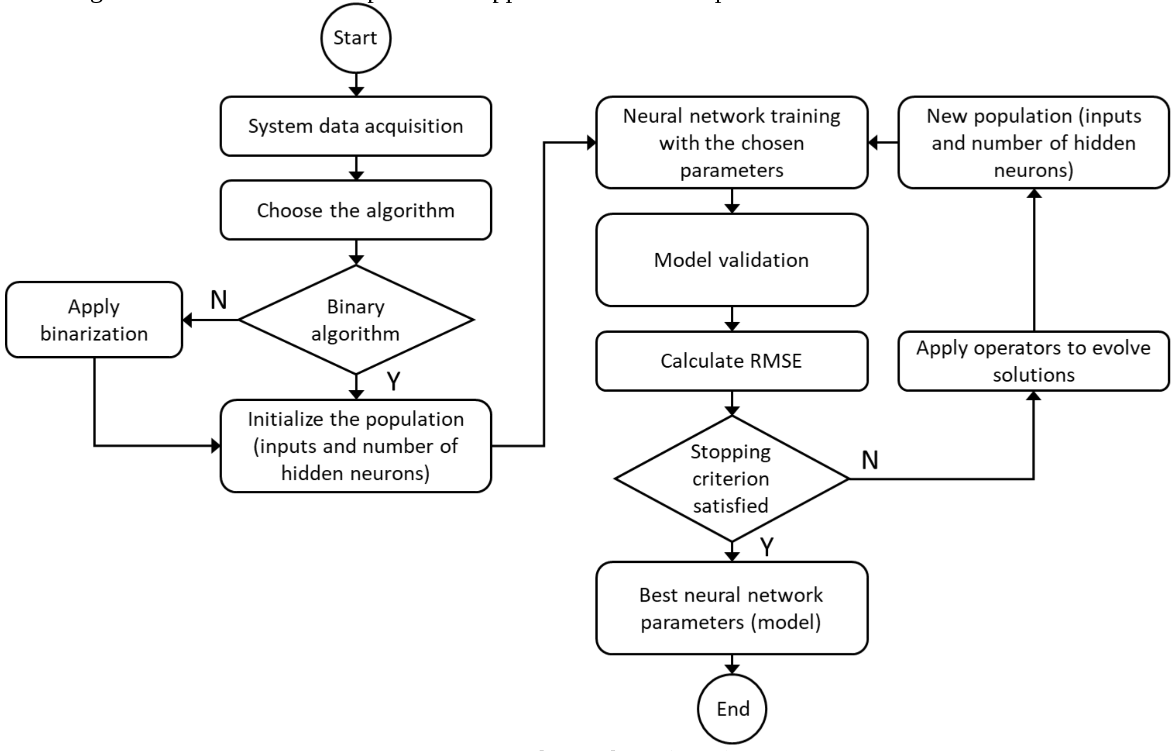
The problem of parameterizing neural networks has some aspects characterized as binary optimization. For example, deciding how many and which inputs the network will have. Evolutionary algorithms are increasingly being applied to solve binary optimization problems. Some of them were originally created to deal with binary problems, such as the Genetic Algorithm, while others were designed to work with continuous variables and require some adaptation to deal with this type of problem, such as Differential Evolution (Dahi; Mezioud; Draa., 2015).

In this work, the binarization used to adapt the Differential Evolution will be: Binary Coding by Transfer Function (TF) *Transfer Function*); Highest Value Priority (GVP) binary encoding *Great Value Priority*); and Binary Angle Modulation (AM) encoding *Angle Modulation*). All the codifications used in the research are detailed in the literature (Crawford *et al.*, 2017).

PROPOSED STRATEGY

The strategy proposed in this work for parameterization of neural networks using evolutionary algorithms consists of the application of some steps, which can be observed in the flowchart in Figure 4.

Figure 4 – Flowchart of the procedure applied for automatic parameterization of neural networks.


Start
System data acquisition
Choose the algorithm
Binary algorithm
N
Y
Apply binarization
Initialize the population (inputs and number of hidden neurons)
Neural network training with the chosen parameters
Model validation
Calculate RMSE
Stopping criterion satisfied
N
Y
Best neural network parameters (model)
End
New population (inputs and number of hidden neurons)
Apply operators to evolve solutions

Source: The Authors (2024).

Initially, experimental data obtained from the studied system should be available. Next, it is necessary to determine which evolutionary algorithm will be used, which will manipulate solutions to perform the automatic parameterization of the neural network. The parameters to be determined in this study are the amount and types of inputs and the amount of neurons in the hidden layer.

If the GA is selected, it has all the structures to determine these parameters. Both the determination of inputs and the choice of the number of neurons can be encoded by a vector in binary format. However, for DE, it will be necessary to apply binarization to determine the inputs, since this evolutionary algorithm is of the continuous type and, in its standard configuration, would only be able to determine the number of neurons.

Considering all possible combinations between the DE algorithm and the binarization (TF, GVP and AM), in addition to including the GA algorithm in the analysis, a total of 4 techniques are obtained to be evaluated in the automatic parameterization process. These techniques are listed in Table 1.

Table 1 - Techniques applied in the automatic parameterization strategy

Metaheuristics	Binary coding	Technique
GA	-	GA
DE	TF	DE-TF
	GVP	DE-GVP
	AM	DE-AM

Source: The Authors (2024).

The next step is to create an initial population of solutions and calculate the suitability of each solution using the objective function. This function will be composed of the neural network being applied in the prediction of the behavior of the system under study, through the stages of training with Extreme Learning Machine and model validation.

The performance of a solution will be based on the validation of the model generated by this solution, with the square root of the mean squared error being applied as the performance index (RMSE, from English *Root Mean Square Error*) calculated using the following equation (Peñaranda; Saavedra-Montes, 2012):

$$RMSE = \sqrt{\frac{\sum_{\mathrm{k}=1}^{N}\left(y(k) - \hat{y}(k)\right)^2}{N}} \tag{4}$$

Where $y(k)$ and $\hat{y}(k)$ are, respectively, the outputs observed and predicted using the validation data, and N is the amount of data used.

From this point, the iterative cycle of evolutionary algorithms begins, where operators are applied to evolve the population of solutions. Each time this cycle is executed, one generation is counted and the same will be repeated until some stop criterion is reached. In this work, the stopping criterion used will be the maximum number of generations.

Once the stop criterion is reached, the algorithm will end the search and the best set of parameters for the neural network will be the one that generates a model with the lowest RMSE. With the completion of this step and selection of the best parameters, a simulation is completed. For the purpose of analyzing the performance of the techniques presented, each simulation will be repeated several times.

To evaluate the performance of each technique, the mean behavior of the convergence curve of the objective function will be initially verified. For this, in each generation the values obtained by all simulations will be taken and the average will be taken.

Another way to evaluate the performance of the techniques will be through the value of the objective function achieved at the end of each simulation. In this case, the minimum, mean, maximum and standard deviation values will be verified among all the simulations performed by each technique.

All the codes used in this work were implemented in the Matlab software and the simulations were run on a laptop with an Intel Core i7 processor and 16GB of RAM (*Random Access Memory*).

CASE STUDIES

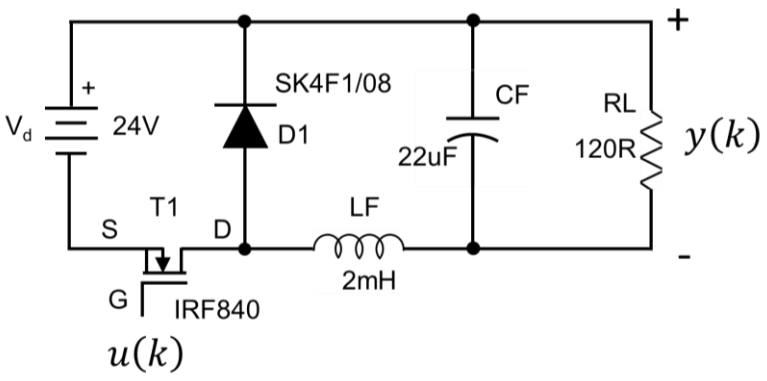
This section presents the test systems that will be used as case studies to validate the proposed strategy.

MODELING A *BUCK CONVERTER*

The first case study of this work consists of using a neural network to model a *Buck* converter. The modeling strategy consists of applying system identification techniques using a single-layer hidden neural network trained with the Extreme Learning Machine algorithm, according to a study conducted by Oliveira and Leandro (2019).

The *Buck* converter that will serve as the object of study was initially discussed in the work of Aguirre, Donoso-Garcia and Santos-Filho (2000) and is often used as a test system in the area of system identification. The *Buck* converter diagram is shown in Figure 5.

Figure 5 – Diagram of the *buck converter*.

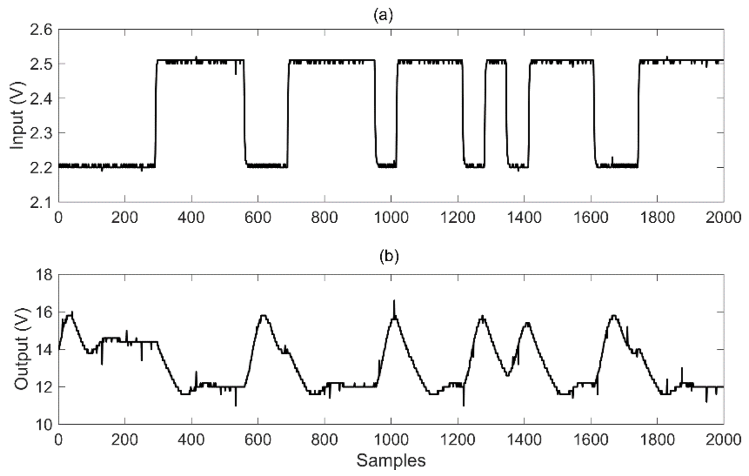


Source: Aguirre, Donoso-Garcia and Santos-Filho (2000).

The load voltage regulation system is not shown in Figure 5. In the experiment performed with this converter, the power supply v_d was kept constant and equal to $24V$. To excite the dynamics of the converter, we opted for an input signal of the PRBS type, limited between $2{,}2V$ and $2{,}5V$, applied to the *Buck* using a Digital/Analog converter (Aguirre; Donoso-Garcia; Santos-Filho, 2000).

In this experiment, the voltage that defines the cyclic ratio of the converter was considered as the input signal, $u(k)$, and the electrical voltage at the output of the converter was adopted as the output signal, $y(k)$. A total of 2000 pairs of samples were collected, which can be seen in Figure 6.

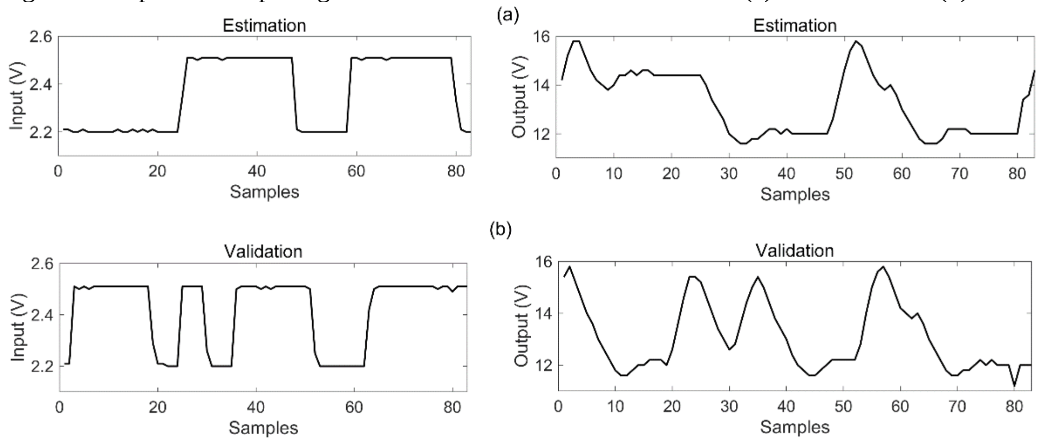
Figure 6 – Data collected from the *buck* converter. (a) input signal and (b) output signal.



Source: Aguirre, Donoso-Garcia and Santos-Filho (2000).

As the data were sampled at a higher rate than necessary, it was decided to apply a working sampling time equal to $T_{s^*} = 120\mu s$, resulting in a total of 166 pairs of samples. In addition, the data were divided into two distinct sets, each with 50% of the 166 sample pairs. The first half of the data was used to estimate the models and the second half was reserved for the validation stage, as shown in Figure 7.

Figure 7 – Input and output signals of the *buck converter*. Estimation (a) and validation (b) data.



Source: The Authors (2024).

The data presented in FIGURE 8 were used by Aguirre, Donoso-Garcia and Santos-Filho to obtain models for the (2000) *buck* converter. These same data were made available on the internet by the authors, which allowed their use in this case study.

PREDICTION OF COMPRESSIVE STRENGTH OF SELF-COMPACTING CONCRETE (SCC) WITH THE ADDITION OF FIBERS

The second case study of this work consists of the use of the neural network to predict the compressive strength of self-compacting concrete (SCC) with the addition of fibers. The literature points to several studies that apply similar approaches (Balasubramaniyam; Padmanabhan, 2022; Gholanjadeh-Chitgar; Berenzian, 2019; Najm; Mohammad; Alzard, 2023; Saha; Prasad; Kumar, 2017; Tavakoli *et al.*, 2014), since the prediction of properties of materials used in civil construction using computational techniques has shown very promising results.

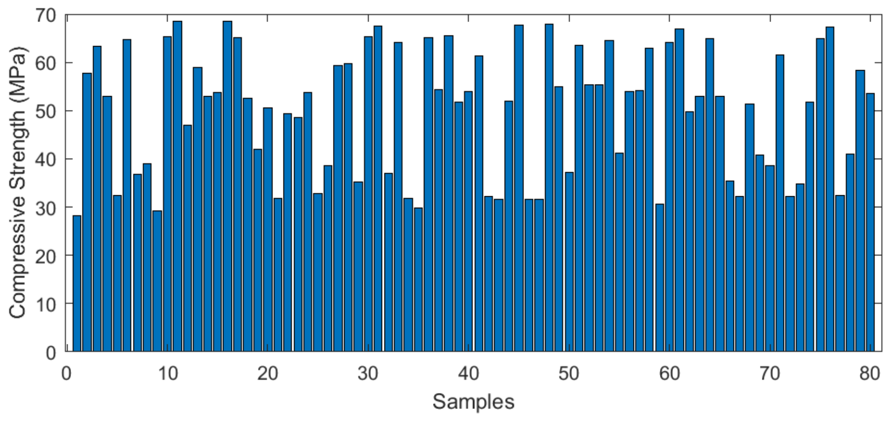
SCC is a material used in civil construction and is characterized by its ability to flow through the heavily reinforced section with the required viscosity without segregation (Balasubramaniyam; Padmanabhan, 2022). In addition, it tends to present greater strengths from the addition of fibers, when compared to concrete without additions.

The database used was experimentally developed by Saha, Prasad and Kumar (2017) containing a total of 99 samples. For the prediction of a single output (compressive strength), nine input parameters were used, which include quantities of cement, fine aggregate (sand), coarse aggregate, fly ash, glass fibers, polypropylene fibers, water, super plasticizer and viscosity modifier additive (VMA).

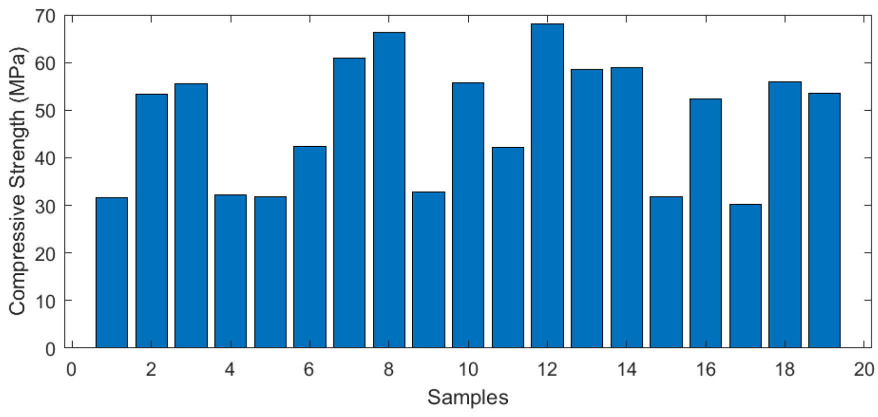
The data were divided into two sets: the training set, with 80% of the data (80 samples), and the test set with 20% of the data (19 samples). Figure 8 shows the output values of these two sets

Figure 8 – SCC compressive strength values. Training (a) and test (b) data.

(a)



(b)



Source: The Authors (2024).

RESULTS

In this section, the results obtained by the application of the strategy proposed in the case studies indicated in section 4 will be presented and discussed.

PARAMETER CONFIGURATION

In order to apply the proposed strategy in the resolution of the case studies presented, it was initially necessary to determine the type of neural network that will be used. The single hidden layer topology was selected, using a sigmoid activation function and trained with the Extreme Learning Machine algorithm.

Next, you must determine the variables that will serve as input to the neural network. In the first case study, since this is a modeling problem, it is necessary to select the set of candidate terms for the buck converter model. This set is shown in Figure 9 (Oliveira; Leandro, 2019). In the second case study, the problem consists in the prediction of the compressive strength of SCC with the addition of fibers, which makes it necessary to indicate which materials will serve as input. In the case of this material, the inputs are: cement, fine aggregate (sand), coarse aggregate, fly ash, glass fibers, polypropylene fibers, water, super plasticizer, viscosity modifier additive (VMA).

Figure 9 – Set of candidate terms for input from the neural network of the buck converter.

1º	2º	3º	4º	5º	6º	7º	8º
$y(k-1)$	$y(k-2)$	$y(k-3)$	$u(k-1)$	$u(k-2)$	$u(k-3)$	$y^2(k-1)$	$y(k-1)\,y(k-2)$

9º	10º	11º	12º	13º	14º
$y(k-1)\,y(k-3)$	$y(k-1)\,u(k-1)$	$y(k-1)\,u(k-2)$	$y(k-1)\,u(k-3)$	$y(k-2)\,y(k-3)$	$y^2(k-2)$

15º	16º	17º	18º	19º	20º
$y(k-2)\,u(k-1)$	$y(k-2)\,u(k-2)$	$y(k-2)\,u(k-3)$	$y^2(k-3)$	$y(k-3)\,u(k-1)$	$y(k-3)\,u(k-2)$

21º	22º	23º	24º	25º	26º	27º
$y(k-3)\,u(k-3)$	$u^2(k-1)$	$u(k-1)\,u(k-2)$	$u(k-1)\,u(k-3)$	$u^2(k-2)$	$u(k-2)\,u(k-3)$	$u^2(k-3)$

Source: Oliveira and Leandro (2019).

In addition to the neural network inputs, the algorithms must select the number of neurons in the hidden layer of the network. This number will vary from 1 to 30 and the ideal value will be determined by the algorithms, by running 10 simulations for each number of neurons tested and choosing the amount that presents the best performance for the network.

For the size of the populations of the evolutionary algorithms, a value of 50 was adopted. The maximum number of generations adopted was the value of 100, which was considered as the criterion for stopping the algorithm.

Specific parameter values of each algorithm were selected heuristically through independent tests. The best values found for these parameters are shown in Table 2.

Table 2 - Parameters of evolutionary algorithms

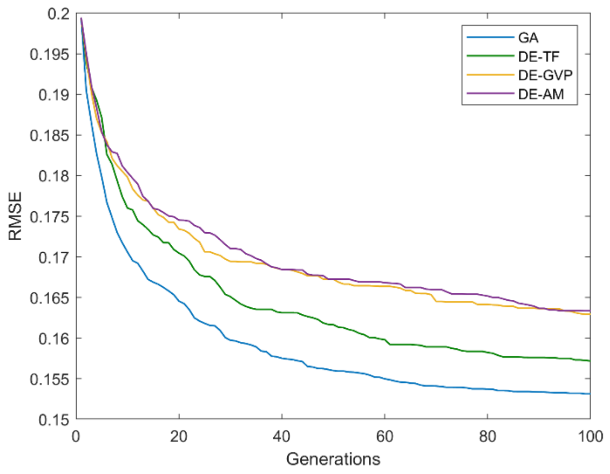
Metaheurística	Parameter	Value
GA	t_{sn}	0,6
	t_c	0,7
	p_m	0,05
DE	F_m	0,7
	C_r	0,6
	Estratégia	DE/rand/1/bin

Source: The Authors (2024).

RESULTS FOR BUCK CONVERTER MODELING

Using the configurations presented in section 5.1, the strategy presented in section 3 was applied to each of the 4 technique options listed in Table 1, and a total of 30 simulations were performed for each technique. The mean convergence curves, resulting from the 30 simulations of each technique, can be seen in Figure 10. On the other hand, the statistical results of the *RMSE* for all techniques are presented in Table 3, and the mean value indicated corresponds to those presented in Figure 10.

Figure 10 – Mean convergence curves of the 4 techniques applied in the modeling of the buck converter.



Source: The Authors (2024).

Table 3 - RMSE values achieved by the techniques in modeling the buck converter

Technique	Minimum	Medium	Maximum	Standard Deviation
GA	**0,14884**	**0,15314**	**0,15640**	**0,0018789**
DE-TF	0,15357	0,15718	0,16403	0,0020830
DE-GVP	0,15381	0,16294	0,16807	0,0035657
DE-AM	0,15605	0,16336	0,16950	0,0033986

Source: The Authors (2024).

Analyzing Figure 10 and Table 3, it is possible to observe that GA was the technique that achieved the best mean performance (lowest mean *RMSE* value) for the buck converter modeling problem, with a value of 0.15314. In second place was the DE-TF technique, with a *mean RMSE* 2.64% higher than that achieved by the GA. The techniques with the worst mean performances were

DE-GVP and DE-AM, with *mean RMSE* 6.40% and 6.67% higher than the GA, respectively. Regarding convergence speeds, all 4 techniques had similar performance, without any stopping at local minimums.

From these considerations, it is possible to verify that, as most of the parameters determined for the network are in binary format (inputs), GA showed a better performance in the determination of such parameters, since this technique is a natively binary algorithm. It was also possible to observe that the choice of binary coding influenced the performance of the DE, with the TF coding being the one that provided the DE with its best average performance.

Looking at the "Minimum" column in Table 3, it is possible to observe the *RMSE* values of the best model found by each technique. These models are a consequence of the best set of parameters (inputs and neurons) that each technique has selected for the neural network. These parameters are shown in Table 4.

Table 4 - Parameters selected by the techniques in the buck converter modeling problem

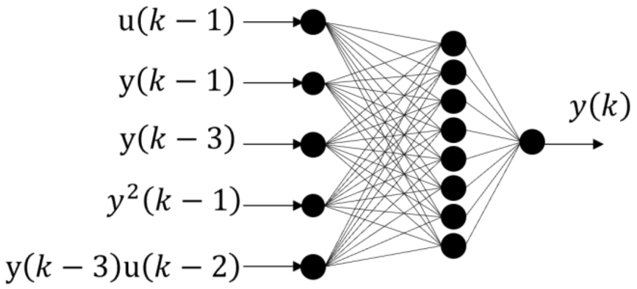
Technique	**Nº of entries**	**Chosen terms**	**Nº of neurons**	**RMSE of the model**
GA	**5**	**1º, 3º, 4º, 7º e 20º**	**8**	**0,14884**
DE-TF	13	1º, 3º, 4º, 6º, 7º, 9º, 10º, 15º, 16º, 17º, 18º, 23º e 24º	9	0,15357
DE-GVP	15	1º, 3º, 4º, 6º, 7º, 9º, 10º, 15º, 18º, 19º, 20º, 21º, 23º, 24º e 27º	15	0,15381
DE-AM	9	1º, 3º, 4º, 7º, 9º, 11º, 13º, 19º e 20º	13	0,15605

Source: The Authors (2024).

Tables 3 and 4 show that the lowest *RMSE* value found was 0.14884, which represents the *RMSE* of the best model found in the entire study. Thus, it can be seen that the GA, in addition to having obtained a better average behavior in terms of convergence, found a set of parameters that generated the best model among all those tested in the simulations. It is also worth mentioning that the set of parameters indicated by the GA results in the network of less complexity, compared to the other parameters indicated by the other techniques in their best scenarios.

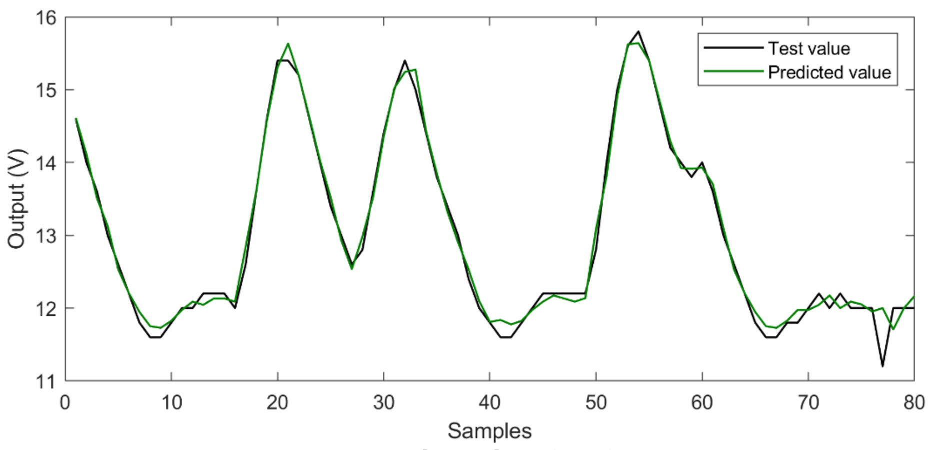
The neural network parameterized by GA has only 8 neurons in the hidden layer and 5 inputs, which receive 3 order terms 1 and 2 order terms 2. This neural network is shown in Figure 11 and the simulation of the model obtained with it is presented in Figure 12.

Figure 11 – Neural network configured with the parameters selected by GA to perform buck converter modeling



Source: The Authors (2024).

Figure 12 – Prediction of a step ahead of the model obtained with the neural network in Figure 11



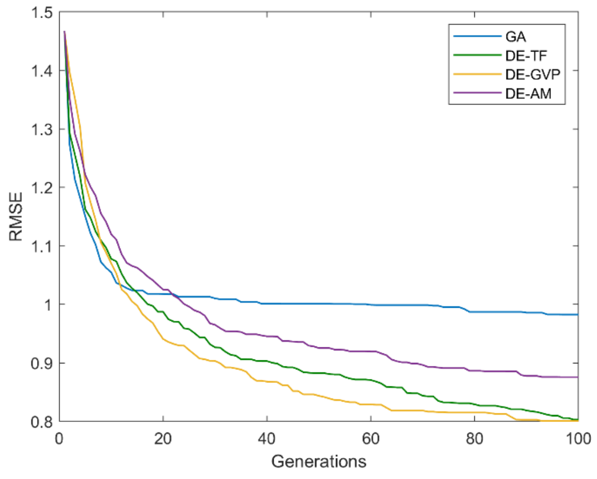
Source: The Authors (2024).

It can be seen from Figure 12 that the model obtained with the neural network presented in Figure 11 represents well the dynamic behavior of the real system, that is, in addition to presenting the lowest *RMSE* among all the models obtained, its validation through simulation also reinforces the quality of the model.

RESULTS FOR THE PREDICTION OF COMPRESSIVE STRENGTH OF SCC WITH THE ADDITION OF FIBERS

The proposed strategy was also applied to the problem of predicting the compressive strength of SCC with the addition of fibers, using the configurations presented in section 5.1, and for each of the 4 options of techniques listed in Table 1, a total of 30 simulations were performed for each technique. The mean convergence curves, resulting from the 30 simulations of each technique, can be seen in Figure 13. The statistical results of the *RMSE* for all techniques are presented in Table 5, and the mean value indicated corresponds to those presented in Figure 13.

Figure 13 – Mean convergence curves of the 4 techniques applied to the problem of predicting the compressive strength of SCC with the addition of fibers.



Source: The Authors (2024).

Table 5 - RMSE values achieved by the techniques applied to the problem of predicting the compressive strength of SCC with added fibers.

Technique	Minimum	Medium	Maximum	Standard Deviation
GA	0,76828	0,98267	1,14150	0,092832
DE-TF	0,73032	0,80330	0,87576	0,039255
DE-GVP	**0,73866**	**0,80071**	**0,88993**	**0,034929**
DE-AM	0,73840	0,87581	0,97690	0,053924

Source: The Authors (2024).

Analyzing Figure 13 and Table 5, it is possible to observe that DE-GVP was the technique that achieved the best mean performance (lowest mean *RMSE value*) for the problem of predicting the compressive strength of SCC, with a value of 0.80071. In second place was the DE-TF technique, with an *average RMSE* only 0.82% higher than that achieved by the DE-GVP. The techniques with the worst mean performance were DE-AM and GA, with *mean RMSE* 9.38% and 22.72% higher than DE-GVP, respectively. Regarding the convergence speeds, the DE-based techniques showed similar performances, without any stopping at local minimums. The GA, on the other hand, showed a higher convergence speed at the beginning of the simulations, but was stuck in many local minimums from the 10th generation onwards.

From these considerations, it is possible to verify that, although most of the parameters determined for the network are in binary format (inputs), in this specific problem the technique that presented the best results was the DE with binary encoding. It was also possible to observe that the choice of binary coding influenced the performance of the DE, with the GVP coding being the one that provided the DE with its best average performance.

Looking at the "Minimum" column in Table 5, it is possible to observe the *RMSE* values of the best model found by each technique. These models are a consequence of the best set of parameters (inputs and neurons) that each technique has selected for the neural network. These parameters are shown in Table 6.

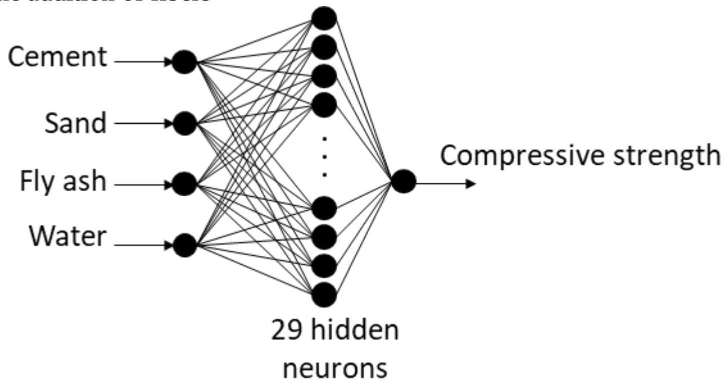
Table 6 - Parameters selected by the techniques in the problem of predicting the compressive strength of SCC with added fibers

Technique	Nº of entries	Entries chosen	Nº of neurons	RMSE of the model
GA	4	cement, sand, fly ash, water	30	0,76828
DE-TF	**4**	**cement, sand, fly ash, water**	**29**	**0,73032**
DE-GVP	5	cement, sand, fly ash, water, VMA	30	0,73866
DE-AM	4	cement, sand, fly ash, water	30	0,73840

Source: The Authors (2024).

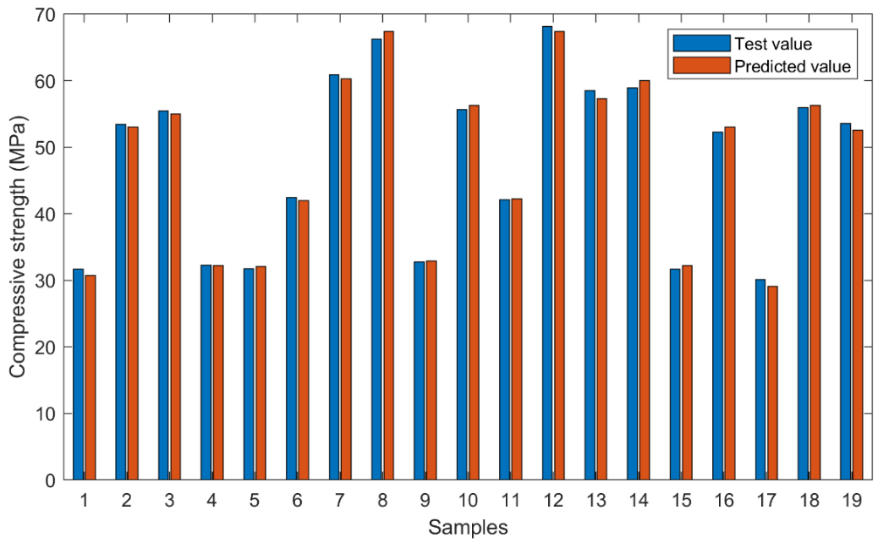
Tables 5 and 6 show that the lowest *RMSE* value found was 0.73032, which represents the *RMSE* of the best model found in the entire study. This model has cement, sand, fly ash and water as inputs and 29 neurons in the occult layer, these parameters being selected by the DE-TF technique. Although this model presented the lowest RMSE value, it was verified that the other techniques also obtained models with parameters similar to this one, both in the inputs and in the number of neurons in the occult layer. The neural network that represents the best model obtained is shown in Figure 14 and the predicted values for the output are shown in Figure 15.

Figure 14 – Neural network configured with the parameters selected by DE-TF to perform the prediction of SCC compressive strength with the addition of fibers



Source: The Authors (2024).

Figure 15 – Prediction of the compressive strength of the model obtained with the neural network in Figure 14



Source: The Authors (2024).

Figure 15 shows that the model obtained with the neural network presented in Figure 14 represents well the behavior of the developed composite, i.e., in addition to presenting the lowest *RMSE* among all the models obtained, the prediction of compressive strength values in values close to the real ones also reinforces the quality of the model.

CONCLUSION

This work presents a comparative study between evolutionary algorithms applied to the problem of automatic parameterization of neural networks used in system identification and prediction of compressive strength of SCC with addition of fibers. Two types of algorithms were used: those that originally manipulate binary solutions and those that are designed to work with continuous values and, therefore, need coding to manipulate solutions in the binary search space.

The evolutionary algorithm of binary solutions applied was the Genetic Algorithm (GA) and the evolutionary algorithm of continuous solutions used was the Differential Evolution (DE). To implement the binary versions of the DE, the encodings by Transfer Function (TF), Highest Value Priority (GVP) and Angle Modulation (AM) were applied.

Two case studies were conducted using single-layer hidden layer neural networks, trained with Extreme Learning Machine, being applied in the modeling of a *buck converter* and in the prediction of compressive strength of SCC with the addition of fibers, both systems presenting data available in the literature.

In the case study that addressed the modeling of the buck converter, the results obtained indicated that the GA presented the best performance, locating the best set of parameters (inputs and neurons) for the network used, generating a model with less complexity and with lower RMSE. Another perception obtained from the results was that the choice of binary coding significantly influenced the performance of the DE, which obtained its best average performance when combined with TF coding. It was also possible to verify that the best model obtained presented good results when submitted to the simulation validation of the one-step forward prediction, adapting well to the dynamic data of the system.

In the other case study, which addressed the prediction of compressive strength of SCC with the addition of fibers, the results showed that the best parameters for the neural network were obtained by the DE-TF technique, generating the model that presented the lowest RMSE. As in the previous case study, it was also possible to observe that the best model obtained presented good results when submitted to validation by comparing the predicted values of compressive strength with the actual values, indicating that the model satisfactorily represented the behavior of the composite presented.

It is noteworthy that the present study was focused on selecting the inputs and the number of neurons of a single hidden layer neural network, not prioritizing, for example, the choice of the activation function, initial weights and biases, training algorithm, among other possible parameters. In addition, the goal was to obtain models with the greatest possible fit to the test data of the systems, without considering other types of characteristics such as model complexity and computational cost.

FUTURE WORK

The possibilities for the continuation of this work involve the following topics, but are not limited to these:

- Perform the parameterization considering other parameters besides the inputs and number of neurons;
- Test the developed approach with other neural network topologies;
- Apply the methods presented to obtain models from other systems;
- Use other evolutionary algorithms to reinforce the conclusions reached;
- Explore other binary encoding methods.

REFERENCES

1. Aguirre, L. A., Donoso-Garcia, P. F., & Santos-Filho, R. (2000). Use of a priori information in the identification of global nonlinear models-a case study using a buck converter. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 47(7), 1081–1085.

2. Balasubramaniam, N., & Padmanaban, I. (2022). Prediction of the strength of self-compacting cementitious mix with glass fibre using machine learning. Journal of Ceramic Processing Research, 23(6), 806–816.

3. Crawford, B., et al. (2017). Putting continuous metaheuristics to work in binary search spaces. Complexity, 2017.

4. Dahi, Z. A. E. M., Meziod, C., & Draa, A. (2015). Binary bat algorithm: on the efficiency of mapping functions when handling binary problems using continuous-variable-based metaheuristics. Springer.

5. De Castro, L. N., & Timmis, J. (2002). Artificial immune systems: a new computational intelligence approach. Springer Science & Business Media.

6. Dorigo, M. (1992). Optimization, learning and natural algorithms. PhD Thesis, Politecnico di Milano.

7. Gaspar-Cunha, A., Takahashi, R., & Antunes, C. H. (2013). Manual de Computação Evolutiva e Metaheurística. Belo Horizonte: Editora UFMG.

8. Gholamzadeh-Chitgar, A., & Berenjian, J. (2019). Elman ANNs along with two different sets of inputs for predicting the properties of SCCs. Computers and Concrete, 24(5), 399–412.

9. Haykin, S. (2001). Redes Neurais: Princípios e prática (2nd ed.). Porto Alegre: Bookman.

10. Holland, J. (1975). Adaptation in natural and artificial systems: an introductory analysis with application to biology, control and artificial intelligence. Control and artificial intelligence.

11. Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. IEEE.

12. Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection (Vol. 1). MIT Press.

13. Lima, I., Pinheiro, C. A. M., & Santos, F. A. O. (2016). Inteligência artificial (Vol. 1). Elsevier Brasil.

14. Najm, O. F., Mohamed, O. A., & Alzard, M. H. (2023). The efficiency of statistical and artificial neural network techniques on evaluating and predicting compressive strength of sustainable SCC with basalt fibers. Materials Today: Proceedings.

15. Oliveira, Á. H. R. D., & Leandro, G. V. (2019). Algoritmos de Otimização Combinados Aplicados à Identificação de Sistemas. XIV Conferência Brasileira de Dinâmica, Controle e Aplicações.

16. Peñaranda, J. R. C., & Saavedra-Montes, A. J. (2012). Dynamic model validation via error indexes. IEEE.

17. Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Royal Aircraft Establishment Library Translation 1122.

18. Saha, P., Prasad, M. L. V., & Kumar, P. R. (2017). Predicting strength of SCC using artificial neural network and multivariable regression analysis. Computers and Concrete, 20(1), 31–38.

19. Schwefel, H-P. (1965). Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. Diploma thesis, Technical Univ. of Berlin.

20. Serre, D. (2002). Matrices: Theory and applications. New York: [s.n.].

21. Silva, I. N. D., Spatti, D. H., & Flauzino, R. A. (2010). Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas (2nd ed.). São Paulo: Artliber.

22. Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, 11(4), 341–359.

23. Tavakoli, H. R., et al. (2014). Prediction of combined effects of fibers and nanosilica on the mechanical properties of self-compacting concrete using artificial neural network. Latin American Journal of Solids and Structures, 11(11), 1906–1923.

24. Wilamowski, B. M., & Irwin, J. D. (2018). Intelligent systems. CRC press.